REMARKS:

In the outstanding Office Action, claims 1-3 and 5-25 were rejected. New claims 26 has been added, thus, in view of the forgoing, claims 1-3, and 5-26 remain pending for which reconsideration is requested. No new matter has been added. The Examiner's rejections are traversed below.

REJECTION UNDER 35 U.S.C. §101:

Claims 1-3 and 5-25 were rejected under 35 U.S.C. §101 as being directed to non-statutory subject matter.

The Examiner claims to find no limitation to a practical application for the claimed method. Independent claims 1, 17, 19, and 20-25 are directed to a method and apparatus for passing algorithms in a dependency graph of a graphics creation process using a computer. Accordingly, the algorithms or functions can be passed between nodes in a graphics system to better control access to needed data and efficient execution of processes involved.

According to MPEP §2106, for process claims related to computer inventions to be statutory, "... a claimed computer related process must either: (A) result in a physical transformation outside the computer for which a practical application in the technological arts is either disclosed in the specification or would have been known to a skilled artisan or (B) be limited to a practical application within the technological arts." Further, MPEP §2106 states that "a claim is limited to a practical application when the method, as claimed, produces a concrete, tangible and useful result."

As recited in independent claims 1, 17, 19, and 20-25, passing a pointer to an algorithm of a first dependency node to a second dependency node allows the second dependency node to execute the algorithm as part of an evaluation of the second dependency node. Accordingly, a concrete, tangible and useful result is achieved because flexibility is afforded by not only passing data with a single value of an attribute with a current state of inputs but also by passing an algorithm that can be used to evaluate the attribute with any desired combinations of inputs (see, State Street Bank & Trust Co. v. Signature Financial Group Inc., 47 U.S.P.Q.2d 1596 (Fed. Cir. 1998)).

It is respectfully submitted that because independent claims 1, 17, and 19 and dependent claims dependent there from satisfy the requirements of 35 USC §101, withdrawal of the rejection is requested.

REJECTION UNDER 35 U.S.C. §103(a):

In the outstanding Office Action, claims 1-3 and 5-25 were rejected under 35 U.S.C. §103(a) as being anticipated by U.S. Patent No. 5,666,296 ('296). The rejection is traversed and reconsideration is respectfully requested.

'296 discusses a method for symbolic evaluation of algorithms with data-dependent control flow by translating a software algorithm with the data-dependent control flow into a data flow graph.

The present application discloses a system for passing algorithms between nodes of a dependency graph in a graphic creation process system where the algorithms are enabled to be executed as an evaluation of other nodes.

The Examiner acknowledges that the '296 method fails to disclose a second dependency node executing an algorithm as part of the second dependency node, however the Examiner asserts that it would have been obvious to one with ordinary skill in the art to pass the algorithm because the '296 method discusses symbolically evaluating each node in succession. In the '296 method, a data flow graph is constructed from a software algorithm where each node contains a set of executable statements (see, column 3, lines 30-34 of '296). Then reverse denominators are computed from the control flow graph indicating a set of nodes through which control from respective node is guaranteed to pass (see, column 3, line 65 through column 4, line 1 of '296). Accordingly, as the Examiner points out, "evaluation of the algorithm is necessary for each node".

The present application does not necessitate evaluation of an algorithm in each node. This is achieved by "passing a pointer to an algorithm associated with a first dependency node to a second dependency node allowing the second dependency node to execute the algorithm" (see, claims 1, 17, 19 and 23-25 of the present application). For example, it is not required that both nodes 6 and 8 in FIG. 2 of the present application be evaluated because f(d)=e is passed to node 8 and executed as part of an evaluation of node 8 without the need to evaluate node 6. This is unlike the '296 method where when an input changes, each node has to be re-evaluated to accommodate for the change because only the result of an algorithm is passed to the next node (see, FIG. 4 and corresponding text of '296). Further, passing f(d)=e to node 8 allows evaluation of f(d)=e in a different context where g(e)=h is used, which yields a completely different result than in f(d)=e in node 6. This effectively rewrites the algorithm executed in node 8 because the algorithm itself is passed to node 8, and may be used to achieve a completely different function than when the algorithm was in node 6.

Further, the present application eliminates the need to evaluate each node when new data

is inputted to provide control and flexible of execution, and does not guarantee that a control will pass through any specific node. For example, the system evaluates "the dependency graph based on the algorithms passed between the nodes" (see, claim 20) or evaluates an algorithm using a passed pointer to the algorithm (see, claim 21), instead of evaluating each node symbolically based on static set of instructions or rules as in the '296 system (see, column 7, lines 56-61 of '296).

As recited in claim 23, the algorithm is "reexecuted" via the second node each time input data of the second node changes. The Examiner uses the same rational used to reject claim 1 to reject claim 23. However, the '296 system merely converts a software program to represent statements of the program (see, column 1, lines 60-67 of '296), and does not teach or suggest reexecuting an algorithm passed from one node to other nodes when input data of the other nodes changes.

Further, the present application as recited in claims 1, 17, 19 and 20-25 passes an algorithm from a first node to a second node where the algorithm of the first node can be executed. Nothing in '296 teaches or suggests this. Each node in '296 is limited to executing its own algorithm and there is no teaching or suggestion of executing an algorithm of another node.

For at least the above discussed reasons, the independent claims 1, 17, 19 and 20-25 and the dependent claims depending from claims 1, 17, 19 and 20-25 are patentably distinct from the '296 method. The dependent claims also recite additional patentably distinguishing features. For example, dependent claim 7 recites a feature where "the algorithm parameter types are identified dynamically as the dependency graph is executed". The symbolic evaluation of '296 yields all possible data values and data types (see, col. 3, lines 8-15 of '296) while in the dependency graph of the present application identifies the parameter types dynamically.

The burden of establishing a prima facie case of obviousness based upon the prior art lies with the Examiner. In re Fritch, 23 U.S.P.Q. 2d 1780, 1783 (Fed. Cir. 1992). According to In re Fritch, the Examiner "... can satisfy this burden only by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings of the references." Per Examiner's own assertion, '296 does not teach a method of allowing a second dependency node to execute an algorithm passed from a first dependency node as part of an evaluation of the second dependency node. And the '296 system is directed to converting a software algorithm to a data flow graph, thus is patentably distinct from the method of the present application where the algorithm is passed from one node and executed in another.

The Applicants respectfully assert that the Examiner has not met the burden of establishing a prima facie case of obviousness and request withdrawal of the outstanding rejections.

NEW CLAIM:

New claim 26 has been added to emphasize an aspect of the present application where a method of evaluating a dependency graph includes "passing a first algorithm associated with a first dependency node to a second dependency node" for allowing the second dependency node having a second algorithm to execute the first algorithm where "the first algorithm is embedded in the second dependency node and executed as part of an evaluation of the second dependency node". This allows the second dependency node to implement the first algorithm of the first node within the second dependency node, and execute the first algorithm as part of an evaluation of the second dependency node, thereby providing an efficient execution of processed in a graphics creation process. The cited reference does not teach or suggest this.

CONCLUSION:

In accordance with the foregoing, new claim 26 has been added, thus claims 1-3 and 5-26 are pending and under consideration.

There being no further outstanding objections or rejections, it is submitted that the application is in condition for allowance. An early action to that effect is courteously solicited.

Finally, if there are any formal matters remaining after this response, the Examiner is requested to telephone the undersigned to attend to these matters.

If there are any additional fees associated with filing of this Amendment, please charge the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 5/26/4

By: _____
J. Randall Beckers
Registration No. 30,358

1201 New York Avenue, NW, Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

CERTIFICATE UNDER 37 CFR 1.8(a)
I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
on May 26, 2004
STAAS & HALSEY
By: Lemmit Afework
Date: 5/26/04